**[Design Patterns - Wikipedia](https://en.wikipedia.org/wiki/Design_Patterns)**

在[软件工程](https://zh.wikipedia.org/wiki/%E8%BB%9F%E9%AB%94%E5%B7%A5%E7%A8%8B" \o "软件工程)中，设计模式（design pattern）是对[软件设计](https://zh.wikipedia.org/wiki/%E8%BB%9F%E4%BB%B6%E8%A8%AD%E8%A8%88" \o "软件设计)中普遍存在（反复出现）的各种问题，所提出的解决方案。这个术语是由[埃里希·伽玛](https://zh.wikipedia.org/wiki/%E5%9F%83%E9%87%8C%E5%B8%8C%C2%B7%E4%BC%BD%E7%91%AA" \o "埃里希·伽玛)（Erich Gamma）等人在1990年代从[建筑设计](https://zh.wikipedia.org/wiki/%E5%BB%BA%E7%AD%91%E8%AE%BE%E8%AE%A1" \o "建筑设计)领域引入到[计算机科学](https://zh.wikipedia.org/wiki/%E8%A8%88%E7%AE%97%E6%A9%9F%E7%A7%91%E5%AD%B8" \o "计算机科学)的。

设计模式并不直接用来完成[代码](https://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BC%8F%E7%A2%BC" \o "代码)的编写，而是描述在各种不同情况下，要怎么解决问题的一种方案。[面向对象](https://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1" \o "面向对象程序设计)设计模式通常以[类别](https://zh.wikipedia.org/wiki/%E7%B1%BB_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6)" \o "类 (计算机科学))或[对象](https://zh.wikipedia.org/wiki/%E7%89%A9%E4%BB%B6_(%E9%9B%BB%E8%85%A6%E7%A7%91%E5%AD%B8)" \o "对象 (计算机科学))来描述其中的关系和相互作用，但不涉及用来完成应用程序的特定类别或对象。设计模式能使不稳定依赖于相对稳定、具体依赖于相对抽象，避免会引起麻烦的紧耦合，以增强软件设计面对并适应变化的能力。

并非所有的软件模式都是设计模式，设计模式特指软件“设计”层次上的问题。还有其他非设计模式的模式，如[架构模式](https://zh.wikipedia.org/wiki/%E6%9E%B6%E6%9E%84%E6%A8%A1%E5%BC%8F" \o "架构模式)。同时，[算法](https://zh.wikipedia.org/wiki/%E6%BC%94%E7%AE%97%E6%B3%95" \o "算法)不能算是一种设计模式，因为算法主要是用来解决计算上的问题，而非设计上的问题。

随着软件开发社群对设计模式的兴趣日益增长，已经出版了一些相关的专著，定期召开相应的研讨会，而且[沃德·坎宁安](https://zh.wikipedia.org/wiki/%E6%B2%83%E5%BE%B7%C2%B7%E5%9D%8E%E5%AE%81%E5%AE%89" \o "沃德·坎宁安)（Ward Cunningham）为此发明了[WikiWiki](https://zh.wikipedia.org/wiki/WikiWiki" \o "WikiWiki)用来交流设计模式的经验。

**发展历史**

建筑师[克里斯托佛·亚历山大](https://zh.wikipedia.org/wiki/%E5%85%8B%E9%87%8C%E6%96%AF%E6%89%98%E4%BD%9B%C2%B7%E4%BA%9A%E5%8E%86%E5%B1%B1%E5%A4%A7" \o "克里斯托佛·亚历山大)在1977/79年编制了一本汇集设计模式的书，但是这种设计模式的思想在建筑设计领域里的影响远没有后来在软件开发领域里传播的广泛。

[肯特·贝克](https://zh.wikipedia.org/wiki/%E8%82%AF%E7%89%B9%C2%B7%E8%B2%9D%E5%85%8B" \o "肯特·贝克)和[沃德·坎宁安](https://zh.wikipedia.org/wiki/%E6%B2%83%E5%BE%B7%C2%B7%E5%9D%8E%E5%AE%81%E5%AE%89" \o "沃德·坎宁安)在1987年，利用克里斯托佛·亚历山大在建筑设计领域里的思想开发了设计模式，并把此思想应用在Smalltalk中的图形用户接口（GUI）的生成中。一年后，[埃里希·伽玛](https://zh.wikipedia.org/wiki/%E5%9F%83%E9%87%8C%E5%B8%8C%C2%B7%E4%BC%BD%E7%91%AA" \o "埃里希·伽玛)在他的[苏黎世大学](https://zh.wikipedia.org/wiki/%E8%8B%8F%E9%BB%8E%E4%B8%96%E5%A4%A7%E5%AD%A6" \o "苏黎世大学)博士毕业论文中开始尝试把这种思想改写为适用于软件开发。与此同时James Coplien 在1989年至1991年也在利用相同的思想致力于C++的开发，而后于1991年发表了他的著作Advanced C++ Programming Styles and Idioms。同年，埃里希·伽玛得到了博士学位，然后去了美国，在那与Richard Helm、Ralph Johnson、John Vlissides 合作出版了《[设计模式：可复用面向对象软件的基础](https://zh.wikipedia.org/wiki/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F%EF%BC%9A%E5%8F%AF%E5%A4%8D%E7%94%A8%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E8%BD%AF%E4%BB%B6%E7%9A%84%E5%9F%BA%E7%A1%80" \o "设计模式：可复用面向对象软件的基础)》（Design Patterns - Elements of Reusable Object-Oriented Software） 一书，在此书中共收录了 23 种设计模式。

这四位作者在软件开发领域里以“四人帮”（英语，Gang of Four，简称GoF）而闻名，并且他们在此书中的协作导致了软件设计模式的突破。有时，GoF也会用于代指《设计模式》这本书。

**表述格式**

表述一个软件设计模式的格式根据作者的不同，划分和名称等都会有所不同。常用的GoF描述模式的格式大致分为以下这些部分：

模式名：每一个模式都有自己的名字，模式的名字使得我们可以讨论我们的设计。

问题：在面向对象的系统设计过程中反复出现的特定场合，它导致我们采用某个模式。

解决方案：上述问题的解决方案，其内容给出了设计的各个组成部分，它们之间的关系、职责划分和协作方式。

别名：一个模式可以有超过一个以上的名称。这些名称应该要在这一节注明。

动机：在哪种情况使用该模式，是本节提供的方案（包括问题与来龙去脉）的责任。

适用性：模式适用于哪些情况、模式的背景等等。

结构：这部分常用类图与交互图阐述此模式。

参与者：这部分提供一份本模式用到的类与对象清单，与它们在设计下扮演的角色。

合作：描述在此模式下，类与对象间的交互。

影响：采用该模式对软件系统其他部分的影响，比如对系统的扩充性、可移植性的影响。影响也包括负面的影响。这部分应描述使用本模式后的结果、副作用、与权衡(trade-off)。

实现：这部分应描述实现该模式、该模式的部分方案、实现该模式的可能技术、或者建议实现模式的方法。

示例：简略描绘出如何以编程语言来使用模式。

已知应用：业界已知的实现示例。

相关模式：这部分包括其他相关模式，以及与其他类似模式的不同。

**分类**

《[设计模式](https://zh.wikipedia.org/wiki/%E8%AE%BE%E8%AE%A1%E8%8C%83%E4%BE%8B" \o "设计范例)》一书原先把设计模式分为创建型模式、结构型模式、行为型模式，把它们通过授权、聚合、诊断的概念来描述。若想更进一步了解关于[面向对象](https://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E7%9A%84%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1" \o "面向对象的程序设计)设计的背景，参考[接口模式](https://zh.wikipedia.org/wiki/%E6%8E%A5%E5%8F%A3%E6%A8%A1%E5%BC%8F" \o "接口模式)、[内聚性](https://zh.wikipedia.org/wiki/%E5%85%A7%E8%81%9A%E6%80%A7_(%E8%A8%88%E7%AE%97%E6%A9%9F%E7%A7%91%E5%AD%B8)" \o "内聚性 (计算机科学))。若想更进一步了解关于面向对象编程的背景，参考[继承](https://zh.wikipedia.org/wiki/%E7%B9%BC%E6%89%BF_(%E8%A8%88%E7%AE%97%E6%A9%9F%E7%A7%91%E5%AD%B8)" \o "继承 (计算机科学))，[接口](https://zh.wikipedia.org/wiki/%E4%BB%8B%E9%9D%A2_(%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88)" \o "接口 (程序设计))，[多态](https://zh.wikipedia.org/wiki/%E5%A4%9A%E5%9E%8B_(%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88)" \o "多态 (面向对象编程))。







